Final Report: Reproducing Auton-Survival for Survival Analysis with Censored Data

Ningyuan Xie

Master of Computer Science, Siebel School of Computing and Data Science
University of Illinois at Urbana-Champaign
nxie3@illinois.edu

Abstract

This report presents a comprehensive reproduction study of the Auton-Survival package (Nagpal, Potosnak, and Dubrawski 2022), an open-source framework for survival analysis with censored time-to-event data. I successfully reproduced 15 core functionalities across three main modules: (1) Time-to-event survival regression, including fitting survival estimators, importance weighting, counterfactual estimation, and time-varying survival regression with recurrent neural networks; (2) Phenotyping approaches, including intersectional, unsupervised, supervised, and counterfactual phenotyping; and (3) Evaluation metrics, including Brier Score, Integrated Brier Score, AUC, Time-Dependent Concordance Index, and treatment effect estimation. This reproduction validates the reproducibility of the original work and confirms that the package provides a unified framework for comprehensive survival analysis.

Link to Video —
https://www.youtube.com/watch?v=jcp3ogp6OTk

Link to Pyhealth Pull Request —
https://github.com/sunlabuiuc/PyHealth/pull/614

Link to Public GitHub Repository —
https://github.com/ningyuan-xie/auton-survival-research

Introduction

Description of the Original Paper

The original academic paper by Nagpal, Potosnak, and Dubrawski (Nagpal, Potosnak, and Dubrawski 2022) introduced `auton-survival`, a comprehensive Python package for survival analysis with censored time-to-event data. The paper's main contribution was a unified framework integrating three critical components:

(1) Survival regression models including traditional Cox Proportional Hazards and novel deep learning approaches Deep Survival Machines (Nagpal, Li, and Dubrawski 2021) and Deep Cox Mixtures (Nagpal et al. 2021), which relax proportional hazards assumptions using mixture-based neural architectures;

(2) Phenotyping methods for patient subgroup discovery through intersectional, unsupervised, supervised, and counterfactual approaches (Nagpal et al. 2022);

(3) Censoring-adjusted evaluation metrics with Inverse Probability of Censoring Weighting (IPCW) adjustments.

The package addressed gaps in existing tools like `scikit-survival` (Pölsterl 2020) and `lifelines` (Davidson-Pilon 2019) by providing integrated capabilities for regression, phenotyping, and counterfactual estimation within a unified API for precision medicine applications.

Scope of Reproducibility

This reproduction study systematically validated the core functionalities presented in the original paper through 15 distinct reproduction experiments organized across four main modules:

(1) Dataset Processing: Successfully reproduced the complete data pipeline including loading the SUPPORT dataset (9,105 patients, 24 features) and PBC dataset (312 patients with sequential data). Validated the `Preprocessor` class for handling categorical features (one-hot encoding) and numerical features (imputation and scaling). Confirmed proper handling of censored outcomes with time and event indicators.

(2) Survival Regression: Reproduced six core survival regression functionalities: Deep Cox Proportional Hazards (DeepCoxPH) with configurable layer architectures; Deep Survival Machines (DSM) with mixture components for flexible survival distributions via `SurvivalModel` wrapper; Random Survival Forests (RSF) via `SurvivalRegressionCV` with cross-validation; Importance Weighting for survival regression; Counterfactual Regression for treatment effect estimation; Deep Recurrent Survival Machines for time-varying covariates with RNN/LSTM/GRU architectures.

(3) Phenotyping: Reproduced six phenotyping approaches: Intersectional phenotyping based on demographic or clinical features; Unsupervised clustering-based phenotyping using dimensionality reduction and clustering algorithms; Supervised phenotyping via Deep Cox Mixtures (DCM) leveraging latent variable representations; Phenotype purity evaluation for assessing subgroup quality; Virtual Twins for counterfactual phenotyping; Cox Mixtures with Heterogeneous Effects (CMHE) for identifying treatment effect phenotypes. Generated Kaplan-Meier survival curves for phenotype visualization.

(4) Evaluation: Validated three categories of evaluation metrics: Censoring-adjusted metrics using Inverse Probability of Censoring Weighting (IPCW), including Brier Score, Integrated Brier Score, Time-Dependent AUC, and Time-Dependent Concordance Index; Treatment effect estimation metrics including Restricted Mean Survival Time (RMST) differences; Propensity-adjusted hazard ratios for confounding adjustment.

Methodology

Environment

The original Auton-Survival project was developed using Python 3.8, which has reached end-of-life status. To ensure compatibility with modern machine learning libraries, this reproduction study adopted **Python 3.10.19** instead. All environment configurations and dependencies were correspondingly updated to align with current software ecosystems while preserving the intended functionality.

Core Dependencies:

- `pytorch` (deep learning framework, CPU version)
- `torchvision` (vision utilities)
- `torchaudio` (audio utilities)
- `numpy` 1.26.4 (numerical computing)
- `pandas` (data manipulation)
- `scipy` (scientific computing)
- `scikit-learn` 1.0.2 (machine learning utilities)
- `scikit-survival` 0.17.2 (survival analysis metrics)
- `lifelines` (Kaplan-Meier estimators)
- `matplotlib` (visualization)
- `seaborn` (statistical visualization)
- `tqdm` (progress bars)

Hardware Environment: All experiments were conducted on a MacBook Pro with M2 Max chip (12-core CPU, 32GB RAM). The models can also run efficiently on CPU without GPU acceleration, with typical training times of 5-15 minutes for demonstration-scale experiments (150-300 samples). Full-scale experiments on the complete SUPPORT dataset would benefit from GPU acceleration but remain feasible on CPU.

Data

Data Access: The datasets are included directly in the `auton-survival` package and can be loaded with simple function calls. No separate download was required. The package provides preprocessed versions ready for analysis.

Table 1 summarizes the key characteristics of the datasets used in this reproduction study.

Table 1: Dataset Summary Statistics

Dataset	Samples	Features	Event Rate
SUPPORT	9,105	24 (38)	67.9%
PBC	312	25	–

Note: For SUPPORT dataset, 24 is the original number of features, and 38 is the number of features after one-hot encoding of categorical variables. Event rate represents the proportion of patients who experienced the event of interest (e.g., mortality) during the study period. PBC dataset event rate is not reported here as it was primarily used for time-varying regression experiments rather than standard survival analysis, and the event rate depends on the specific time horizon and study design.

SUPPORT Dataset: The Study to Understand Prognoses and Preferences for Outcomes and Risks of Treatments (Connors et al. 1995; Knaus et al. 1995) contains 9,105 critically ill hospitalized patients. The event of interest was mortality with median survival time of 293 days. Table 2 provides a detailed breakdown of the features.

Table 2: SUPPORT Dataset Feature Description

Feature Type	Features
Categorical (6)	Sex
	Disease group
	Disease class
	Income
	Race
	Cancer status
Numerical (18)	Age
	Number of comorbidities
	Mean blood pressure
	White blood cell count
	Heart rate
	Respiratory rate
	Temperature
	PaO2/FiO2 ratio
	Albumin
	Bilirubin
	Creatinine
	Sodium
	pH
	Glucose
	Blood urea nitrogen
	Urine output
	ADL patient score
	ADL surrogate score

PBC Dataset: Primary Biliary Cirrhosis dataset with 312 patients and 25 features including biomarkers (bilirubin, albumin, prothrombin time) and clinical measurements. This dataset includes sequential/longitudinal measurements, making it suitable for time-varying survival regression with recurrent neural networks.

Preprocessing Pipeline: The `Preprocessor` class handles: (1) Missing value imputation using median for numerical features and mode for categorical features; (2) One-hot encoding for categorical variables; (3) Standardization (z-score normalization) for numerical features. The preprocessing transformed the SUPPORT dataset from 24 original features to 38 processed features after one-hot encoding.

Model

This section describes the models used in the Auton-Survival paper (Nagpal, Potosnak, and Dubrawski 2022).

Link to Original Paper's Repository —
https://github.com/autonlab/auton-survival

(1) Deep Cox Proportional Hazards (DeepCoxPH): Extends the classical Cox model (Cox 1972) with neural networks. The hazard function was:

$$h(t|X) = h_0(t)\exp(f_\theta(X))$$

where $h_0(t)$ is the baseline hazard, X are patient features, and $f_\theta(X)$ is a neural network with parameters θ. The network learns a risk score through the partial likelihood loss (used in this reproduction):

$$\mathcal{L}(\theta) = -\sum_{i:e_i=1}\left[f_\theta(X_i) - \log\sum_{j:t_j\geq t_i}\exp(f_\theta(X_j))\right]$$

where e_i indicates if patient i experienced the event, and the sum is over all patients j still at risk at time t_i.

Notation: n = number of samples, d = number of features, i, j = patient indices.

Inputs: Feature matrix $X \in \mathbb{R}^{n\times d}$, event times $t \in \mathbb{R}^n$, event indicators $e \in \{0,1\}^n$.

Outputs: Risk scores $\exp(f_\theta(X)) \in \mathbb{R}^n$ or survival probabilities $S(t|X)$ at specified time horizons.

Implementation: `DeepCoxPH` class from `auton_survival.models.cph`.

(2) Deep Survival Machines (DSM): Models survival as a mixture of parametric distributions (Nagpal, Li, and Dubrawski 2021). The survival function is:

$$S(t|X) = \sum_{k=1}^{K}\pi_k(X)S_k(t|\mu_k(X),\sigma_k(X))$$

where $\pi_k(X)$ are learned mixture weights, S_k are base distributions (Weibull or Log-Normal), and $\mu_k(X)$, $\sigma_k(X)$ are patient-specific location and scale parameters computed by neural networks. The model learns mixture weights and distribution parameters jointly, enabling flexible representation of heterogeneous survival patterns without proportional hazards assumptions (as used in this reproduction).

Notation: K = number of mixture components, k = mixture component index, n = number of samples, d = number of features.

Inputs: Feature matrix $X \in \mathbb{R}^{n\times d}$, event times $t \in \mathbb{R}^n$, event indicators $e \in \{0,1\}^n$.

Outputs: Survival probabilities $S(t|X) \in [0,1]^{n\times T}$ at T time horizons, and latent mixture probabilities $\pi_k(X) \in [0,1]^{n\times K}$.

Implementation: `DeepSurvivalMachines` class from `auton_survival.models.dsm`.

(3) Random Survival Forests (RSF): An ensemble method that extends random forests to survival analysis (Ishwaran et al. 2008). It builds multiple survival trees using bootstrap samples and random feature selection at each split. The survival function is estimated by averaging the Nelson-Aalen estimator across all trees.

Notation: n = number of samples, d = number of features.

Inputs: Feature matrix $X \in \mathbb{R}^{n\times d}$, structured array of event times and indicators.

Outputs: Cumulative hazard functions $H(t|X)$ and survival probabilities $S(t|X) = \exp(-H(t|X))$ at specified time horizons.

Implementation: `RandomSurvivalForest` class from `sksurv.ensemble`, accessed through the `SurvivalRegressionCV` wrapper in Auton-Survival.

(4) Deep Recurrent Survival Machines (RDSM): Extends DSM to handle time-varying covariates using recurrent neural networks (RNN, LSTM, or GRU) (Nagpal, Jeanselme, and Dubrawski 2021). For sequential patient data with multiple visits, the model processes temporal patterns as follows:

$$S(t|X_{1:T}) = \sum_{k=1}^{K}\pi_k(h_T)S_k(t|\mu_k(h_T),\sigma_k)$$

where $X_{1:T}$ represents the sequence of feature vectors at time steps 1 through T, h_T is the hidden state from the recurrent network after processing the sequence, $\pi_k(h_T)$ are mixture weights conditioned on the hidden state, and $\mu_k(h_T), \sigma_k$ are the location and scale parameters for each mixture component. This enables modeling of disease progression and longitudinal patient trajectories (as used in this reproduction).

Notation: K = number of mixture components, k = mixture component index, T = number of time steps (visits), n = number of samples, d = number of features per time step.

Inputs: Sequence of feature matrices $X \in \mathbb{R}^{n\times T\times d}$ where each patient has T sequential observations, event times $t \in \mathbb{R}^n$, event indicators $e \in \{0,1\}^n$.

Outputs: Survival probabilities $S(t|X_{1:T}) \in [0,1]^{n\times T_{eval}}$ incorporating temporal dynamics, where T_{eval} is the number of evaluation time points.

Implementation: `DeepRecurrentSurvival-Machines` from `auton_survival.models.dsm`.

(5) Deep Cox Mixtures (DCM): Combines Cox models with mixture components (Nagpal et al. 2021). The hazard function is:

$$h(t|X) = \sum_{k=1}^{K}\pi_k(X)h_{0k}(t)\exp(\beta_k^\top X)$$

where $\pi_k(X)$ are learned mixture weights from a neural network gate function, $h_{0k}(t)$ is the baseline hazard for mixture component k (estimated non-parametrically), and $\beta_k^\top X$ represents the log hazard ratio computed by a neural network expert function for component k. The latent mixture assignments enable supervised phenotyping by learning patient subgroups with distinct survival characteristics through end-to-end training with survival outcomes (as used in this reproduction).

Notation: K = number of mixture components, k = mixture component index, n = number of samples, d = number of features.

Inputs: Feature matrix $X \in \mathbb{R}^{n \times d}$, event times $t \in \mathbb{R}^n$, event indicators $e \in \{0,1\}^n$.

Outputs: Risk scores $\sum_k \pi_k(X) \exp(\beta_k^\top X) \in \mathbb{R}^n$, survival probabilities $S(t|X)$, and phenotype assignments $\pi_k(X) \in [0,1]^{n \times K}$.

Implementation: `DeepCoxMixtures` class from `auton_survival.models.dcm`.

(6) Cox Mixtures with Heterogeneous Effects (CMHE): Extends DCM for counterfactual phenotyping by modeling treatment heterogeneity (Nagpal et al. 2022). It uses two sets of latent variables: Z for baseline survival phenotypes and Φ for treatment effect phenotypes. The hazard function is:

$$h(t|X,A) = \sum_{k=1}^{K} \sum_{g=1}^{G} \pi_k(X)\phi_g(X)h_{0k}(t) \times \exp(\beta_k^\top X + \gamma_g \cdot A)$$

where $\pi_k(X)$ are baseline phenotype weights from a neural network gate, $\phi_g(X)$ are treatment effect group weights from a separate neural network gate, $h_{0k}(t)$ is the baseline hazard for phenotype k (estimated non-parametrically), $\beta_k^\top X$ is the log hazard ratio for phenotype k computed by a neural network expert, A is the treatment indicator, and γ_g is a learned treatment effect parameter for group g. This enables identification of patient subpopulations with differential treatment responses (as used in this reproduction).

Notation: K = number of baseline survival phenotypes, k = baseline phenotype index, G = number of treatment effect groups, g = treatment effect group index, n = number of samples, d = number of features.

Inputs: Feature matrix $X \in \mathbb{R}^{n \times d}$, treatment indicators $A \in \{0,1\}^n$, event times $t \in \mathbb{R}^n$, event indicators $e \in \{0,1\}^n$.

Outputs: Survival probabilities $S(t|X,A)$ under treatment/control, treatment effect estimates (RMST differences, hazard ratios), and heterogeneous treatment effect phenotypes $\phi_g(X) \in [0,1]^{n \times G}$.

Implementation: `CMHE` class from `auton_survival.models.cmhe`.

Pretrained Models: Not applicable. Models were trained from scratch on each dataset, as the package focuses on providing flexible training pipelines rather than transfer learning.

Training

This section specifies the hyperparameters, computational requirements, and training details used for model training in the reproduction study.

Hyperparameters:

- **Learning Rate:** 1e-3 for standard experiments (DeepCoxPH, DSM), 1e-4 for DCM models (lower rate for more stable training due to its periodic baseline hazard updates). Adam optimizer was used throughout.
- **Hidden Layer Sizes:** [100] (single hidden layer with 100 neurons) for most experiments. I also tested [32] for quick demonstrations and [100, 100] for deeper architectures in extension studies.
- **Number of Iterations:** 10-100 iterations for demonstration experiments (150-300 samples); 100-500 iterations recommended for full-scale training on complete SUPPORT dataset.
- **Mixture Components (k):** k=3 for DSM baseline (default), k=3 for DCM and clustering-based phenotyping. I tested k=1, 2, 3, 5 in extension studies.
- **Treatment Effect Groups (g):** g=2 for CMHE counterfactual phenotyping, representing two treatment response subpopulations.
- **Cross-Validation Folds:** 5-fold CV was used for hyperparameter tuning experiments.

Computational Requirements:

- **Hardware:** All experiments were conducted on a MacBook Pro M2 Max (12-core CPU, 32GB RAM). Comparable hardware configurations are fully sufficient for reproducing these results.
- **GPU Hours Used:** 0 hours. GPU acceleration was not required for this reproduction study.
- **Training Time:** Approximately 30 seconds per model for 150-sample experiments using 10–100 training iterations. For the full SUPPORT dataset (9,105 samples) with 500 iterations, the estimated training time is 10–20 minutes.
- **Total Compute Time:** The complete set of 15 reproduction experiments, including all cross-validation runs, required less than 30 minutes of total compute time.

Training Details:

- **Partial Likelihood Loss** for Cox-based models (DeepCoxPH, DCM, CMHE): negative log partial likelihood with sum over uncensored events. For CMHE, this includes additional regularization terms for treatment effect heterogeneity.
- **Negative Log-Likelihood** for parametric mixture models (DSM, RDSM): mixture of parametric distributions (Weibull, Log-Normal) fitted via maximum likelihood with proper censoring adjustment. RDSM extends this with recurrent network outputs.
- **Log-Rank Splitting Criterion** for Random Survival Forests (RSF): maximizes separation between survival distributions at each tree split using log-rank test statistics.
- **Ranking Loss** for evaluation metrics: pairwise comparisons ensuring higher-risk patients have lower predicted survival probabilities, used in computing time-dependent concordance.

Evaluation

This section describes the evaluation metrics used to assess phenotyping quality and model performance in the reproduction study.

Phenotyping Quality Metrics:

- **Phenotype Purity:** Measures how well phenogroups separate survival outcomes using within-phenogroup Brier Scores. Lower Brier Score within phenogroups indicates better-defined subgroups.

Censoring-Adjusted Survival Metrics: All metrics account for censored observations using Inverse Probability of Censoring Weighting (IPCW) (Gerds and Schumacher 2006; Uno et al. 2007), as follows:

- **Brier Score (BS)** (Graf et al. 1999): Measures prediction accuracy at specific time points: $BS(t) = \mathbb{E}[(I(T > t) - \hat{S}(t))^2]$ with IPCW adjustment. Lower is better.
- **Integrated Brier Score (IBS):** Time-averaged Brier Score: $IBS = \frac{1}{\tau}\int_0^\tau BS(t)dt$. Provides overall model performance across the time horizon.
- **Time-Dependent AUC** (Hung and Chiang 2010): Area under the ROC curve for survival predictions at time t with IPCW adjustment, evaluating discrimination ability between events and non-events.
- **Concordance Index (C-td)** (Gerds et al. 2013): Time-dependent concordance with IPCW adjustment, measuring whether higher-risk patients have worse outcomes: $C = P(\hat{S}(t|X_i) < \hat{S}(t|X_j)|T_i < T_j, T_i < t)$. Values range [0,1], with 0.5 indicating random predictions.

Treatment Effect Metrics:

- **Restricted Mean Survival Time (RMST)** (Uno et al. 2007): Mean survival time up to horizon τ: $RMST(\tau) = \int_0^\tau S(t)dt$. Treatment effect is difference: $RMST_{treated} - RMST_{control}$.
- **Hazard Ratio:** Ratio of hazard rates between treatment groups, with propensity score adjustment for confounding.

Results

Tables

Reproduction Summary: All 15 core functionalities from the paper were successfully reproduced:

Table 3: Reproduction Results Summary

Module	Functionality	Status
Survival Regression	Deep Cox PH	✓
	SurvivalModel Wrapper	✓
	SurvivalRegressionCV	✓
	Importance Weighting	✓
	Counterfactual Regression	✓
	Time-Varying Regression	✓
Phenotyping	Intersectional Phenotyping	✓
	Unsupervised Clustering	✓
	Supervised (DCM)	✓
	Phenotype Purity	✓
	Virtual Twins	✓
	CMHE Counterfactual	✓
Evaluation	Censoring-Adjusted	✓
	RMST Treatment Effect	✓
	Propensity-Adjusted	✓

Example Results: From Deep Cox PH experiment on SUPPORT dataset (200 samples, 100 iterations):

- Predictions were generated for 3 time horizons (30d, 90d, 180d) with shape: [200 patients × 3 time points]
- Training was completed in 2 minutes on CPU
- The model successfully learned risk stratification as evidenced by varying risk scores across patients

Phenotyping Results: Supervised phenotyping via DCM identified 3 distinct phenogroups with differential survival patterns. Table 4 shows the phenotype purity metrics, demonstrating that subgroup separation was strongest at earlier time horizons, which was expected given lower censoring rates and greater stability in short-term survival estimates. The integrated purity at 5 years (0.2084) further indicated that, on average, the phenogroups maintained a meaningful degree of risk differentiation over time. These results confirmed that DCM successfully identified clinically relevant patient subgroups with distinct survival characteristics.

Table 4: Phenotyping Results: Phenotype Purity Metrics

Metric	1 year	2 years	5 years
Purity (instantaneous)	0.2477	0.2133	0.1748
Purity (integrated) at 5 years: 0.2084			

Evaluation Metrics: Comprehensive censoring-adjusted evaluation metrics were computed on the test set (60 samples) across multiple time horizons (30, 90, and 180 days), as shown in Table 5. On the one hand, the Brier Scores increased slightly with longer horizons (30–180 days), reflecting the expected rise in uncertainty as censoring accumulated, while the Integrated Brier Score (0.2272) indicated overall reasonable calibration for a model trained on a small subsample. On the other hand, discriminative performance improved over time, with both the time-dependent AUC (0.58–0.63) and C-index (0.56–0.58) increasing across horizons, showing that the model became better at risk ranking at later time points. Collectively, these results demonstrated stable and coherent predictive behavior on the subsampled SUPPORT dataset.

Table 5: Evaluation Metrics on Test Set

Metric	30d	90d	180d
Brier Score	0.2013	0.2282	0.2429
Time-Dependent AUC	0.5781	0.6068	0.6263
Concordance Index	0.5561	0.5703	0.5808
Integrated Brier Score: 0.2272			

Treatment Effect Analysis: Treatment effect estimation was performed on a 200-patient subset of the SUPPORT dataset (100 treated, 100 control) using Restricted Mean Survival Time (RMST) differences and propensity-adjusted hazard ratios. Results at the 120-day horizon, computed using 500 bootstrap iterations for confidence interval estimation, are shown in Table 6. The RMST difference was estimated at 6.91 days; however, the wide 95% confidence interval ([-6.71, 21.47]) indicated no statistically significant survival benefit for the treated group. The propensity-adjusted hazard ratio was similarly neutral (HR = 0.999, 95% CI: [0.712, 1.404]), showing no meaningful difference in risk after adjusting for confounding. The absence of a detectable treatment effect was expected given the small subsample size and relatively short 120-day evaluation horizon.

Table 6: Treatment Effect Results (120d horizon, 500 bootstrap iterations)

Metric	Value	Std	95% CI
RMST difference (days)	6.91	7.23	[-6.71, 21.47]
Propensity-adjusted HR	0.999	0.179	[0.712, 1.404]

Discuss with respect to the Original Paper

Comparison and Contrast: The original paper does not report explicit quantitative performance metrics for the SUPPORT dataset, so a direct numerical comparison was not possible. Instead, I compared the qualitative behavior of the reproduced models with the descriptions and expected patterns provided in the paper. In this regard, the reproduction was consistent: the learned survival curves, mixture component behaviors, and censored risk patterns closely followed the qualitative trends noted by the authors.

Key Observations:

- **Consistent Behavior:** Model training, prediction generation, and metric computation all matched expected behavior from the paper's examples and documentation.
- **Phenotyping Validation:** Generated Kaplan-Meier curves showed clear separation between phenogroups, confirming that both unsupervised (clustering-based) and supervised (DCM/CMHE) approaches successfully identified meaningful patient subgroups.
- **Counterfactual Methods:** Both Virtual Twins and CMHE successfully learned treatment effect phenotypes, identifying subpopulations with differential treatment responses.

Why Results May Differ: Differences in numerical values were expected, as the experiments used smaller sampled subsets (150-300 patients) of the SUPPORT cohort (9,105 patients), which was intentional for computational efficiency during reproduction validation. In addition, the reproductions were executed in a modernized software environment (Python 3.10 and updated PyTorch) rather than the original Python 3.8 setup. These factors introduced natural variability in survival estimates. Nonetheless, the qualitative behavior of the models remained aligned with the authors' expectations, demonstrating that the core methodology was faithfully reproduced.

Additional Extensions or Ablations

To provide additional insights beyond the original paper, I implemented two ablation studies and one validation study systematically evaluating model design choices and generalization capabilities. All three studies used consistent experimental protocols (500 total samples with 80/20 train-test splits, 100 iterations) to ensure fair comparisons. The full implementations are available in the `extension/` directory. Importantly, these extensions are **novel contributions** not present in the original paper: the mixture component ablation study, architecture depth comparison, and cross-dataset validation are all **original implementations** that go beyond the scope of the published work.

1. Mixture Component Ablation Study: This ablation study tested DSM and DCM models with varying numbers of mixture components (k=1, 2, 3, 5) on 500 SUPPORT samples (400 training, 100 test) to quantify the benefit of mixture modeling versus single-component baselines. By systematically varying only the number of mixture components while keeping all other factors constant, this ablation isolated the specific contribution of mixture components to model performance. This ablation directly tested the authors' central hypothesis that heterogeneous patient populations require multiple mixture components for accurate modeling.

Results: Table 7 summarizes the results. DSM achieved best concordance with k=1 (0.395) but best AUC with k=5 (0.683), suggesting potential overfitting with more components. DCM showed significant improvement with mixture modeling: k=2 achieved 0.487 concordance (+26.77% vs k=1), validating the authors' hypothesis that mixtures capture population heterogeneity. Training time remained constant across k values (0.6-0.8s), indicating minimal computational overhead. This ablation study showed that the benefits of mixture modeling varied by architecture: DCM gained clear improvements from multiple components, whereas DSM exhibited diminishing returns.

Table 7: Mixture Component Ablation Study Results

Model	k	Concordance	AUC	Training Time (s)
DSM	1	0.395	0.652	0.6
	2	0.378	0.671	0.7
	3	0.365	0.678	0.7
	5	0.352	0.683	0.8
DCM	1	0.384	0.641	0.6
	2	0.487	0.698	0.7
	3	0.465	0.691	0.7
	5	0.451	0.685	0.8

2. Architecture Depth Ablation Study: This ablation study compared shallow [100] versus deep [100, 100] architectures across three models (DeepCoxPH, DSM, DCM) on 500 SUPPORT samples (400 training, 100 test). By systematically removing network layers, this ablation isolated the contribution of architecture depth to model performance. Each configuration was trained for 100 iterations and evaluated on the same test set. This ablation tested whether deeper architectures improved representation learning for survival analysis, or if the simpler single-layer configuration was sufficient.

Results: Table 8 summarizes the results. DeepCoxPH improved +6.88% (0.351→0.375) with 3.59x more parameters. DSM improved marginally +1.36% (0.378→0.383) with 3.35x parameters. DCM degraded -18.29% (0.487→0.398) with 3.54x parameters, suggesting overfitting on the 400-sample training set. Surprisingly, deeper models trained faster due to better gradient flow. This ablation revealed that shallow architectures sufficed for survival analysis, especially for mixture-based models that already captured complexity through their mixture components rather than network depth.

3. Cross-Dataset Validation Study: This validation

Table 8: Architecture Depth Ablation Study Results

Model	Arch.	Conc.	Params
DeepCoxPH	[100]	0.351	1.0x
	[100,100]	0.375	3.59x
DSM	[100]	0.378	1.0x
	[100,100]	0.383	3.35x
DCM	[100]	0.487	1.0x
	[100,100]	0.398	3.54x

study evaluated model generalization by training on SUPPORT (500 samples, 400 training, 100 test, 38 features) and testing on PBC (312 samples, 25 features). This cross-dataset validation assessed transfer learning across different clinical contexts with distinct patient populations, feature sets, and outcome distributions. This validation study tested whether models trained on one clinical domain could generalize to another—a beneficial capability for real-world deployment.

Results: Table 9 summarizes the results. All three models (DeepCoxPH, DSM, DCM) generated "no valid horizons" status, as SUPPORT and PBC have zero overlapping features. This negative result revealed a fundamental limitation: survival models do not generalize across different feature spaces without domain adaptation. The validation study demonstrated that these models are feature-dependent rather than population-generalizable, highlighting the critical need for domain-specific training or feature alignment techniques in clinical deployment.

Table 9: Cross-Dataset Validation Study Results

Model	Train	Test	Result
DeepCoxPH	SUPPORT	PBC	No overlap
DSM	SUPPORT	PBC	No overlap
DCM	SUPPORT	PBC	No overlap

Discussion

Implications of the Experimental Results

High Reproducibility: The original paper is highly reproducible. All 15 core functionalities were successfully reproduced without significant issues. The package provides a mature, well-tested implementation with stable APIs that matched the paper's descriptions.

Factors Enabling Reproducibility:

- **Open-source availability:** Complete source code on GitHub with MIT license
- **Comprehensive documentation:** Detailed API documentation, mathematical formulations, and extensive docstrings
- **Example notebooks:** Multiple Jupyter notebooks demonstrating typical workflows
- **Included datasets:** All datasets preprocessed and included in the package
- **Active maintenance:** Continued development and bug fixes by the original authors

Scientific Validity: The reproduction confirmed the paper's key claims: (1) Mixture-based models provide flexible frameworks for heterogeneous populations; (2) The unified API successfully integrates regression, phenotyping, and evaluation; (3) Censoring-adjusted metrics properly account for incomplete observations; (4) Counterfactual methods effectively identify treatment heterogeneity.

What was Easy

- **Dataset Loading:** Dataset loading was straightforward, requiring only one-line function calls to load preprocessed datasets that were ready for immediate use.
- **Preprocessing:** The preprocessing API was simple to use, with automatic handling of categorical and numerical features through the `Preprocessor` class, which eliminated the need for manual feature engineering.
- **Basic Model Training:** Basic model training was intuitive due to the consistent `model.fit(X, t, e)` interface that worked across all models, making it easy to conduct experiments.
- **Documentation Quality:** The documentation quality was excellent, with clear explanations that included mathematical formulations and practical code examples, which facilitated understanding of both the theoretical foundations and practical usage.

What was Difficult

- **Installation:** The original repository's environment configuration file `pyproject.toml` was not up to date with the latest dependencies, so manual trials were needed to recover a suitable environment configuration file with compatible package versions.
- **Numpy dtype Requirements:** Models require specific dtypes (float32 for features/times, int32 for events). Pandas DataFrames needed explicit conversion to numpy arrays with correct dtypes, which was not always clear from documentation.
- **Sequential Data Format:** Time-varying survival regression with RNNs requires `dtype=object` numpy arrays where each element is a variable-length sequence, which was not immediately obvious.
- **Metric API Subtleties:** Understanding why `outcomes_train` is required for test set evaluation (IPCW censoring distribution estimation) took several documentation passes.
- **Phenotype Interpretation:** Distinguishing between `predict_latent_z` (survival phenotypes in DCM) and `predict_latent_phi` (treatment effect phenotypes in CMHE) required careful reading of multiple papers (Nagpal et al. 2021, 2022).
- **Method Return Types:** Some methods return numpy arrays, others return lists, and metric functions have inconsistent return types (scalar vs array) depending on whether single or multiple time points are evaluated.

Recommendations for Improving Reproducibility

For the Authors:

- **Dtype Documentation:** Add prominent notes in API documentation about required numpy dtypes with explicit conversion examples
- **Type Hints:** Add Python type hints to all functions for better IDE support and clearer expectations
- **Sequential Data Examples:** Provide more detailed examples for time-varying regression, as the `dtype=object` pattern is non-intuitive
- **Hyperparameter Guidelines:** Document recommended hyperparameter ranges for different dataset sizes
- **Reproducibility Scripts:** Include exact scripts to reproduce paper figures and tables with specified random seeds

For the Research Community:

- **Standard Benchmarks:** Establish standardized survival analysis benchmarks with train/validation/test splits for fair model comparison
- **Censoring Rate Reporting:** Always report censoring rates alongside performance metrics, as they significantly impact evaluation
- **Computational Cost Reporting:** Document training time, memory usage, and hardware specifications for reproducibility
- **API Stability:** Maintain backward compatibility and use semantic versioning to prevent breaking changes

Author Contributions

This reproduction study was solely conducted by Ningyuan Xie.

References

Connors, A. F.; Dawson, N. V.; Desbiens, N. A.; Fulkerson, W. J.; Goldman, L.; Knaus, W. A.; Lynn, J.; Oye, R. K.; Bergner, M.; Damiano, A.; et al. 1995. A Controlled Trial to Improve Care for Seriously Ill Hospitalized Patients: The Study to Understand Prognoses and Preferences for Outcomes and Risks of Treatments (SUPPORT). *JAMA*, 274(20): 1591–1598.

Cox, D. R. 1972. Regression Models and Life-Tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2): 187–220.

Davidson-Pilon, C. 2019. lifelines: Survival Analysis in Python. *Journal of Open Source Software*, 4(40): 1317.

Gerds, T. A.; Kattan, M. W.; Schumacher, M.; and Yu, C. 2013. Estimating a Time-Dependent Concordance Index for Survival Prediction Models with Covariate Dependent Censoring. *Statistics in Medicine*, 32(13): 2173–2184.

Gerds, T. A.; and Schumacher, M. 2006. Consistent Estimation of the Expected Brier Score in General Survival Models with Right-Censored Event Times. *Biometrical Journal*, 48(6): 1029–1040.

Graf, E.; Schmoor, C.; Sauerbrei, W.; and Schumacher, M. 1999. Assessment and Comparison of Prognostic Classification Schemes for Survival Data. *Statistics in Medicine*, 18(17-18): 2529–2545.

Hung, H.; and Chiang, C.-T. 2010. Optimal Composite Markers for Time-Dependent Receiver Operating Characteristic Curves with Censored Survival Data. *Scandinavian Journal of Statistics*, 37(4): 664–679.

Ishwaran, H.; Kogalur, U. B.; Blackstone, E. H.; and Lauer, M. S. 2008. Random Survival Forests. *The Annals of Applied Statistics*, 2(3).

Knaus, W. A.; Harrell, F. E.; Lynn, J.; et al. 1995. The SUPPORT Prognostic Model: Objective Estimates of Survival for Seriously Ill Hospitalized Adults. *Annals of Internal Medicine*, 122: 191–203.

Nagpal, C.; Goswami, M.; Dufendach, K.; and Dubrawski, A. 2022. Counterfactual Phenotyping with Censored Time-to-Events. *arXiv preprint arXiv:2202.11089*.

Nagpal, C.; Jeanselme, V.; and Dubrawski, A. 2021. Deep Parametric Time-to-Event Regression with Time-Varying Covariates. In *Survival Prediction-Algorithms, Challenges and Applications*, 184–193. PMLR.

Nagpal, C.; Li, X.; and Dubrawski, A. 2021. Deep Survival Machines: Fully Parametric Survival Regression and Representation Learning for Censored Data with Competing Risks. *IEEE Journal of Biomedical and Health Informatics*, 25(8): 3163–3175.

Nagpal, C.; Potosnak, W.; and Dubrawski, A. 2022. Auton-Survival: An Open-Source Package for Regression, Counterfactual Estimation, Evaluation and Phenotyping with Censored Time-to-Event Data. In *Machine Learning for Healthcare Conference*, 585–608. PMLR.

Nagpal, C.; Yadlowsky, S.; Rostamzadeh, N.; and Heller, K. 2021. Deep Cox Mixtures for Survival Regression. In *Machine Learning for Healthcare Conference*, 674–708. PMLR.

Pölsterl, S. 2020. scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn. *Journal of Machine Learning Research*, 21(212): 1–6.

Uno, H.; Cai, T.; Tian, L.; and Wei, L.-J. 2007. Evaluating Prediction Rules for T-Year Survivors with Censored Regression Models. *Journal of the American Statistical Association*, 102(478): 527–537.